Assignment 2 - Report

Building Neural Networks and CNNs

APPENDIX:

PART I	1. Dataset Statistics 2. Preprocessing 3. Visualization 4. Neural Network 5. Performance Metrics
PART II	1. Neural Network Optimization 2. Result Visualization
PART III	1. Dataset Overview 2. Visualization 3. CNN Model summary 4. Result Analysis
PART IV	1. VGG-13 Model 2. Architecture 3. Result Analysis

TEAM INFO:

TEAMMATE_1: Dhiraj Surendra Patil - 50604210 - dhirajsu@buffalo.edu
TEAMMATE_2: Anirudh Mhaske - 50604524 - amhaske2@buffalo.edu

Part 1

Dataset Statistics

- The dataset we have is having 7 features named ‘f1’, ‘f2’, ‘f3’, ‘f4’, ‘f5’, ‘f6’, ‘f7’, and also have target column at the end having binary ‘0’ and ‘1’ values for classification.

- The number of samples in the dataset is 766 as each feature column has exactly 766 rows provided.

- We can get the statistics of each feature column with the help of describe() function, which provides us with mean, median, standard deviation and relevant statistical information.

- Below are the provided statistical review of each feature column:

```
count    766.000000
mean       3.845953
std        3.373062
min        0.000000
25%        1.000000
50%        3.000000
75%        6.000000
max       17.000000
Name: f1, dtype: float64
```

```
count    766.000000
mean     120.882507
std       31.935995
min        0.000000
25%       99.000000
50%      117.000000
75%      140.000000
max      199.000000
Name: f2, dtype: float64
```

```
count    766.000000
mean      69.118799
std       19.376901
min        0.000000
25%       62.500000
50%       72.000000
75%       80.000000
max      122.000000
Name: f3, dtype: float64
```

```
count    766.000000
mean      20.515666
std       15.967341
min        0.000000
25%        0.000000
50%       23.000000
75%       32.000000
max       99.000000
Name: f4, dtype: float64
```

```
count    766.000000
mean      79.986945
std      115.335259
min        0.000000
25%        0.000000
50%       34.000000
75%      127.750000
max      846.000000
Name: f5, dtype: float64
```

```
count    766.000000
mean      31.998172
std        7.893111
min        0.000000
25%       27.300000
50%       32.000000
75%       36.600000
max       67.100000
Name: f6, dtype: float64
```

```
count    766.000000
mean       0.471843
std        0.331421
min        0.078000
25%        0.244000
50%        0.372500
75%        0.625500
max        2.420000
Name: f7, dtype: float64
```

- We can also check for missing or null values in each feature column or in the whole dataset with the help of isnull() function and summation of it would give us an exact count of null values. In our scenario we didn't have any null value in any feature and target column:

```
f1        0
f2        0
f3        0
f4        0
f5        0
f6        0
f7        0
target    0
dtype: int64
```

Preprocessing

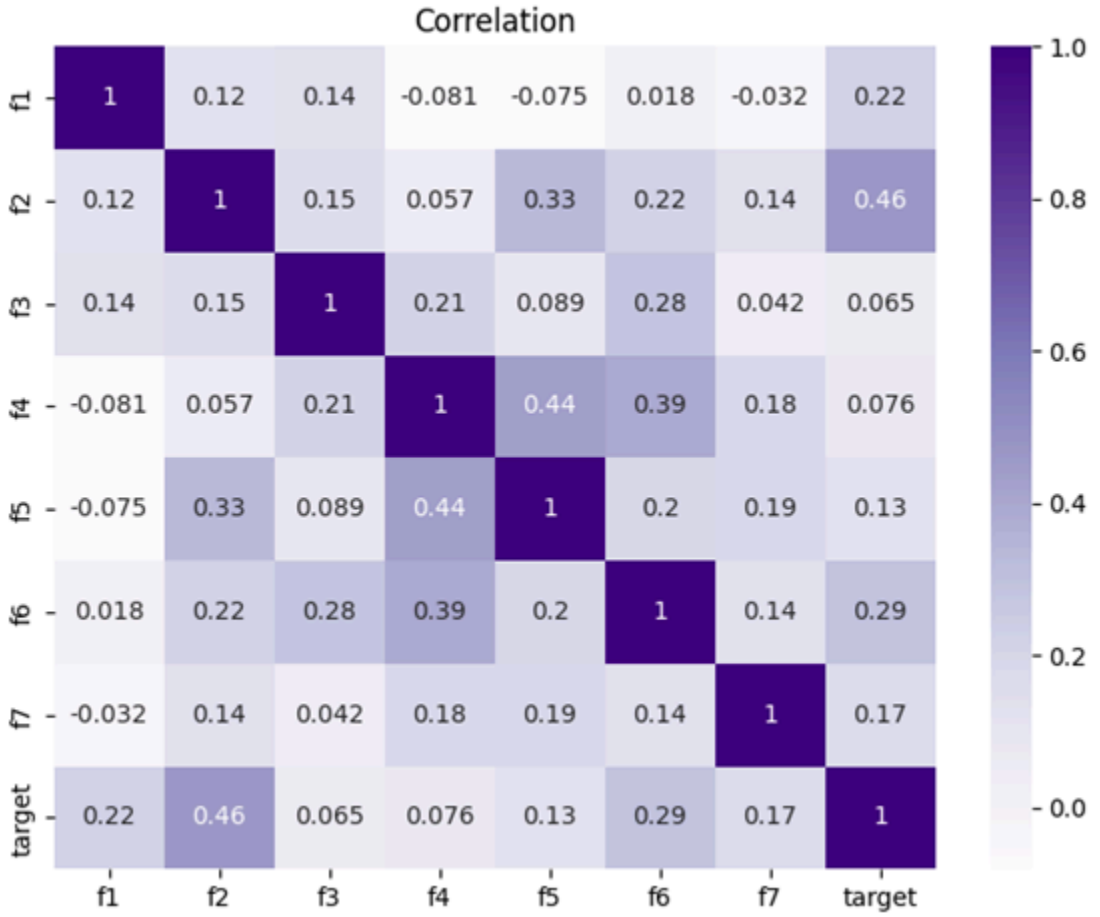
-As part of preprocessing we checked for unique values in each of the feature columns with the help of unique() function. After we see any value that is inappropriate, which in our case is our feature column has characters 'a' , 'c', 'd'….. in each feature column so we need to remove this string or character data in order to make the dataset consistent.

- we can do this with the help of replace() function where we can specify which values in feature columns should be replaced specifically. In our scenario we will replace these characters with NaN values in order to get a common replace factor when we further replace or impute these values.

- After replacing with NaN we will now compute it with mode of the feature column rather than dropping the entire sample or row.

Visualization

1. **Correlation Heatmap:**



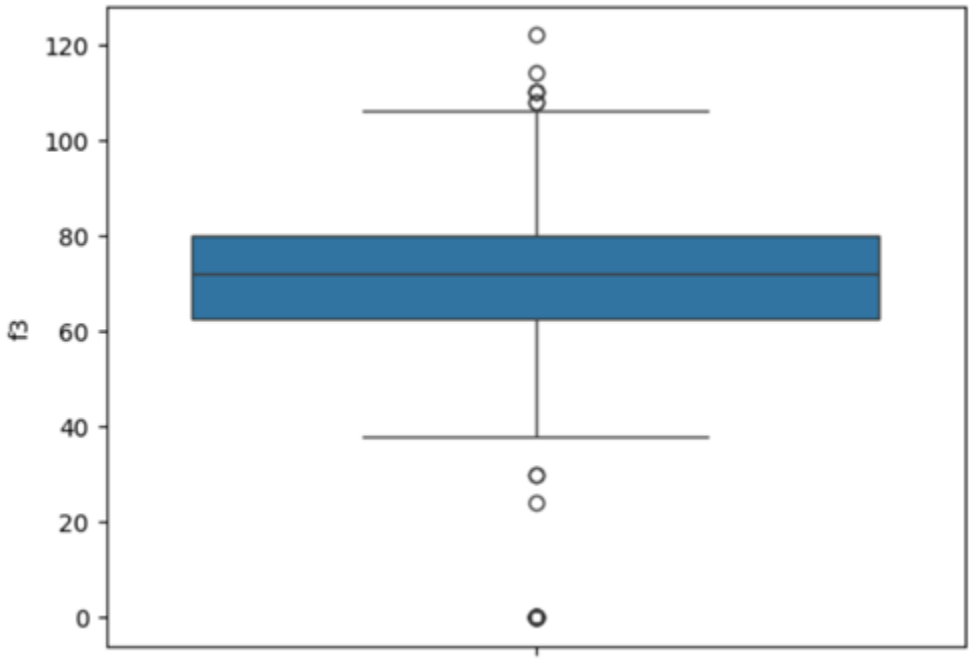
- The correlation heatmap shows extreme correlation diagonally because each feature is paired with itself diagonally, so we see strong correlation values that are '1' when we see it diagonally.

- If we observe the correlation feature 1 'f1' having lease correlation with feature and feature 5 ('f4', 'f5') as we see the values in correlation matrix are negative. Far the value from '1' less it is correlated so negative values means it has very less correlation.

- On the other hand we could see a strong correlation between feature 4('f4') with feature 5 and feature 6('f5', 'f6') as it has values close to '1' showing the strong correlation.

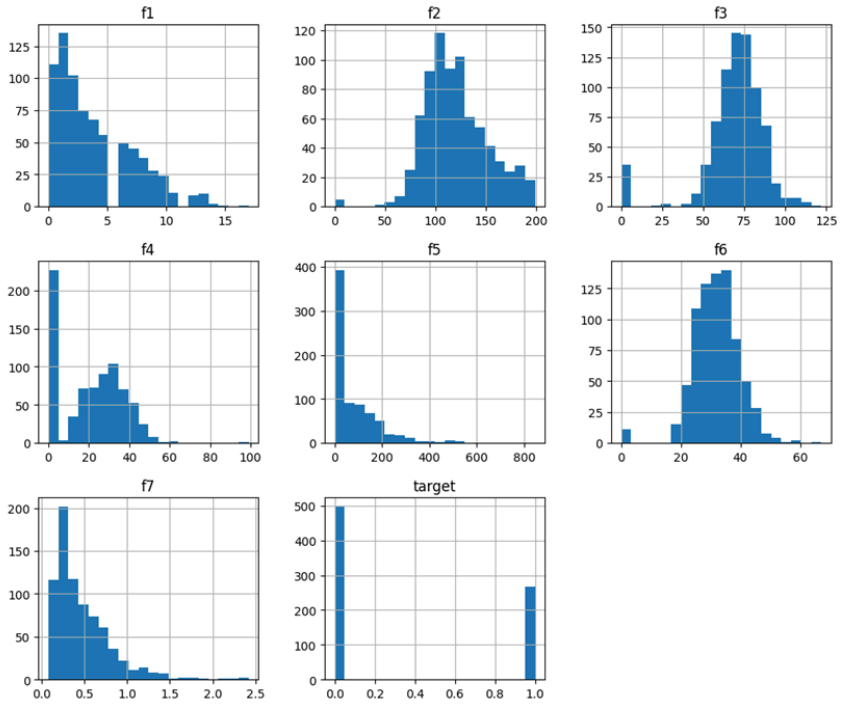
2. Box Plot:

- We have used a box plot on a feature column ‘f3’ to see the quartile range and the outliers present in that particular feature column.



3. Histogram:

- we have executed hist plots for each feature column and target column also , we could see subplots below for each feature.



Neural Network Summary

- The neural network we are choosing has 3 hidden layers, 1 output layer and ReLU as an activation function.

- The output layer has 1 output with a sigmoid function for probability distribution.

- layer 1 has 32 neutrons and 7 features passed, on it batch normalization is then applied and then for activation function we are using ReLU.

- layer 2 has 64 neutrons and 32 input passes, on it batch normalization is then applied and then for activation function we are using ReLU.

- layer 3 has 128 neurons and 64 inputs passed, on it batch normalization is then applied and then for activation function we are using ReLU.

- An output layer having 2 neurons as we are doing binary classification. After layer 3 we have applied dropout.

Performance Metrics

- we are observing the test loss derived from loss function is CrossEntropyLoss() 255.89 and test accuracy nearing to 75%, it is fluctuating between 70-75%.

- We have got the precision value of 0.57 which is almost 57%, this suggests that our model was correct while getting the prediction 575% of the time.

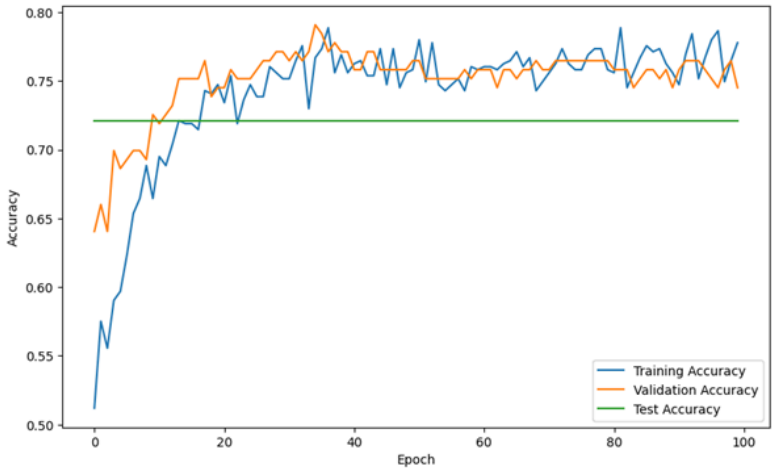
- We have observed the recall value 0.83 which means it was able to successfully predict positive values and missed out on the remaining 17% of the values.

- F1 score is reported to be 67.69% which is almost 70% , so we can say the model is doing good while predicting for positive values.

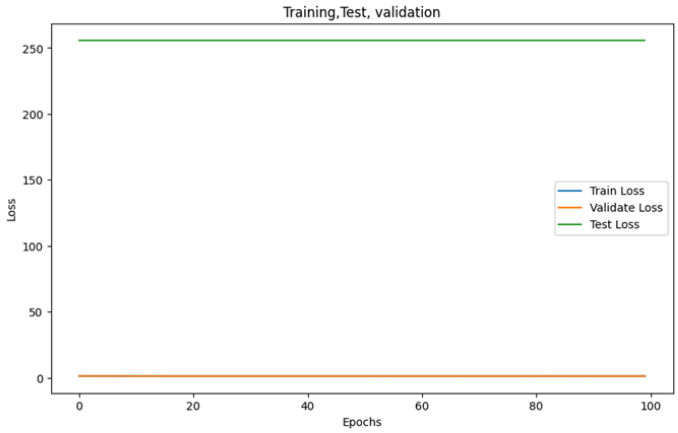
Performance graph:

1. The graph compares train, test and validation accuracy over the epochs in the below graph.

We could see the test accuracy to be constant but there is fluctuation in training and validation accuracy, but still it moves alongside.



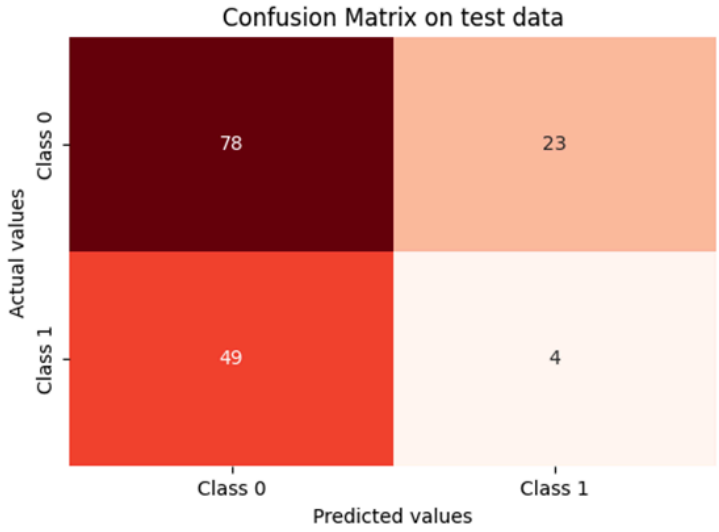
2. In the train, test and validation loss graph we see almost all losses to be constantly moving where the test loss is moving constantly near above 250.



3. Confusion Matrix:

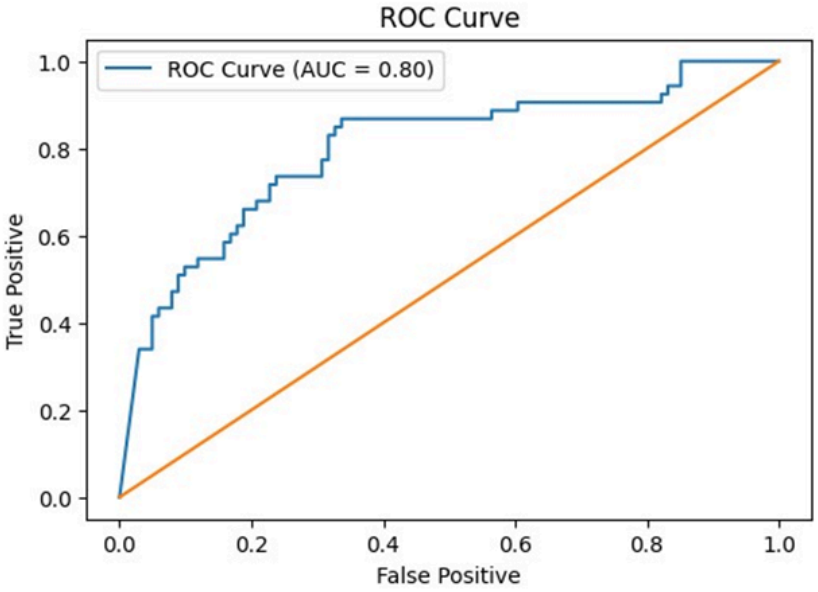
The confusion matrix is observed as below:

Actual values we see 78 values true positive, 23 values false positive, 49 to be false negative and 4 values to be true negative.



4. ROC curve:

Below is the ROC curve we observed:


ROC Curve
ROC Curve (AUC = 0.80)
True Positive
False Positive
0.0
0.2
0.4
0.6
0.8
1.0

Part 2

Neural Network Optimization

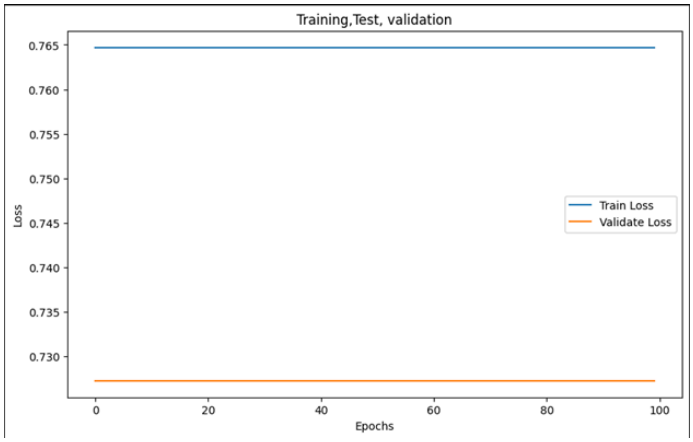
	Value	Test Accuracy
Setup 1	Adam optimizer	74
Setup 2	Updated learning rate to 0.001 and epoch 200.	75.32
Setup 3	Updated number layer	62.33

- Batch Normalization: We have did batch normalization with batch size 32 and we have received the accuracy for it is 72.72%

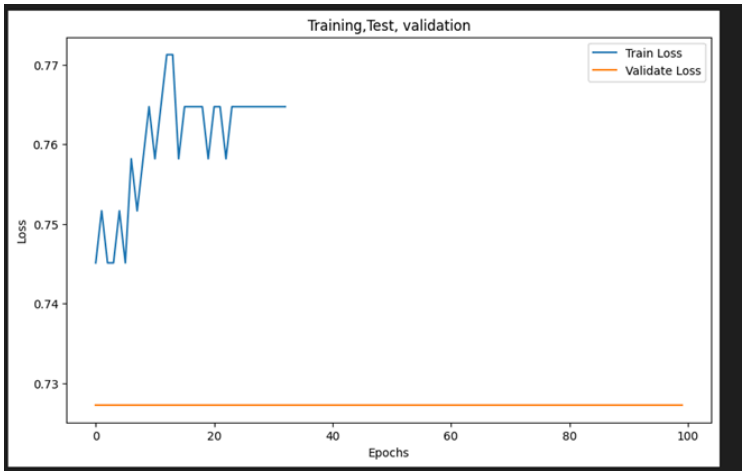
- Gradient Accumulation: The we have implemented gradient accumulation, we just split the batches of samples into smaller batches. For this we received the accuracy to be 72%.

- Early Stopping: In this we stop the model from running when there is no improvement in the model performance, we were running with the no of epochs to be 100. The accuracy we got here is almost 73%.

- for gradient accumulation we have observed below graph for train loss and validation loss:

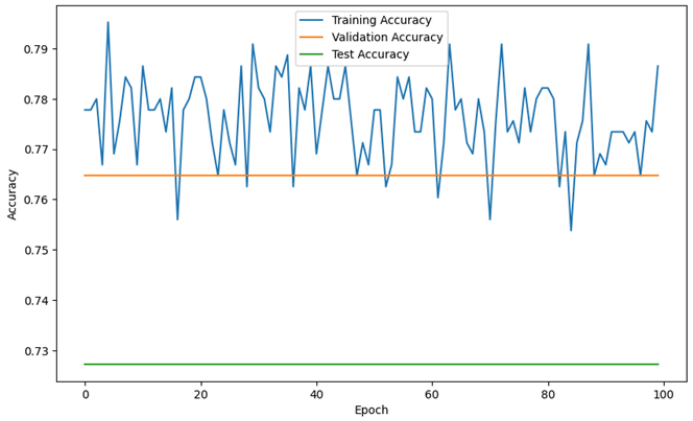


- for early stopping we have observed below graph for train loss and validation loss, where the validation loss seems to be constant:



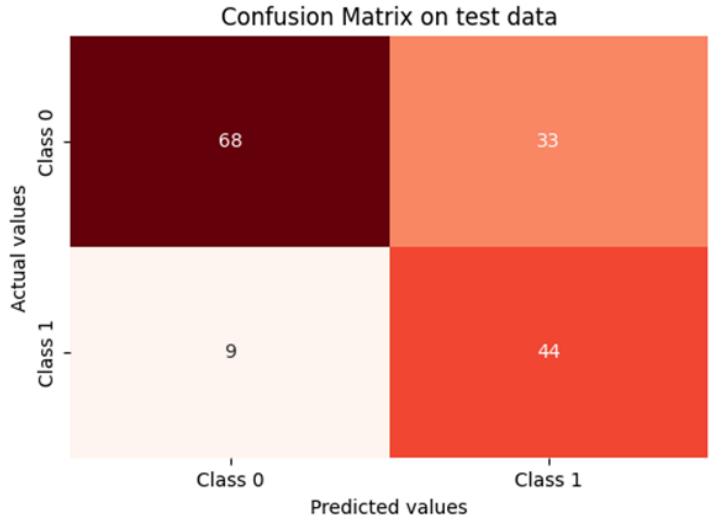
Also for the accuracy we have received the graph as below:

Where we see training accuracy to be fluctuating but test and validation accuracy to be constant at the same time.



Confusion matrix:

We have observed below confusion matrix on test data:



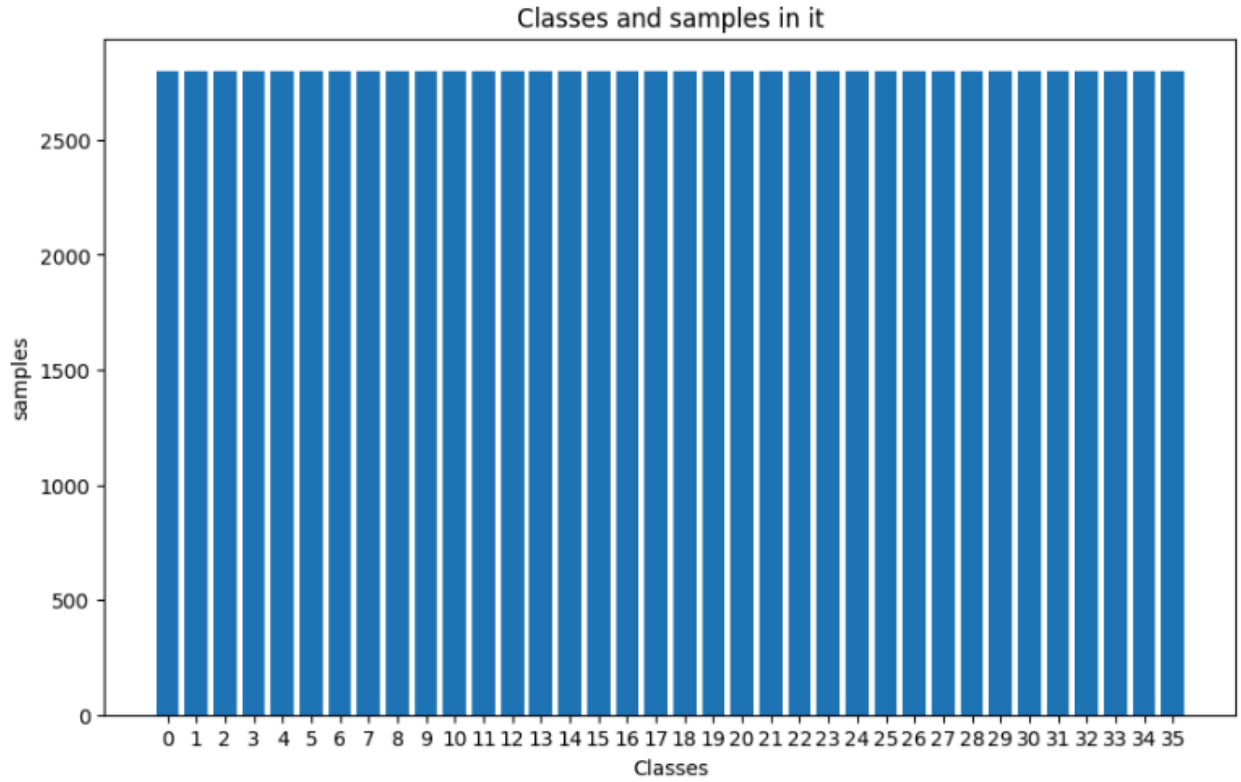
Part 3

1.Brief overview of your dataset:

- The given dataset is similar to MNIST dataset but in MNIST dataset we have 60,000 samples but here in our dataset we have 100,800 sample images provided.
- There are 36 different classes, which are 0-9 numbers having 10 classes and a-z alphabets having 26 classes which combines to 36 distinct classes.
- In our cnn_dataset folder there are subfolders of each class and specifically each class has 2800 samples or images.
- Each of the 2800 images in each class are of standard same size which is 28x28.

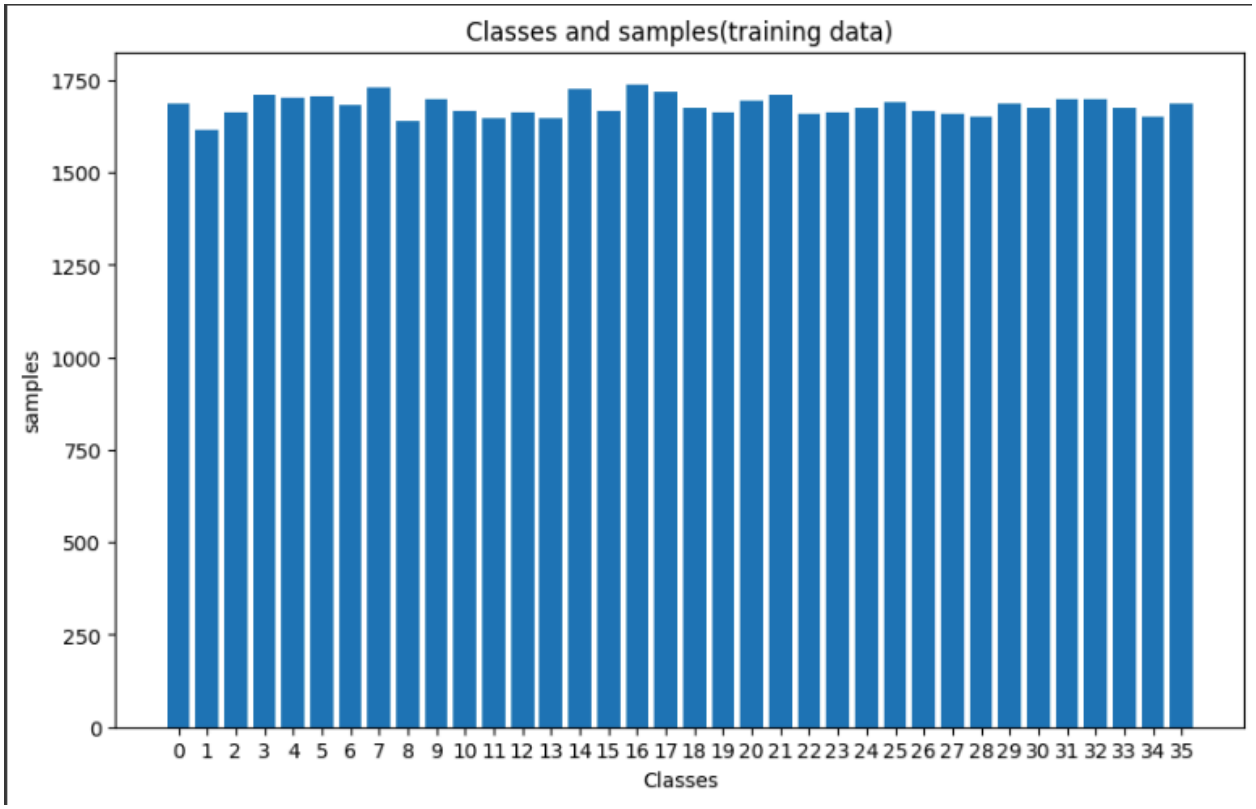
2.Visualization:

A.



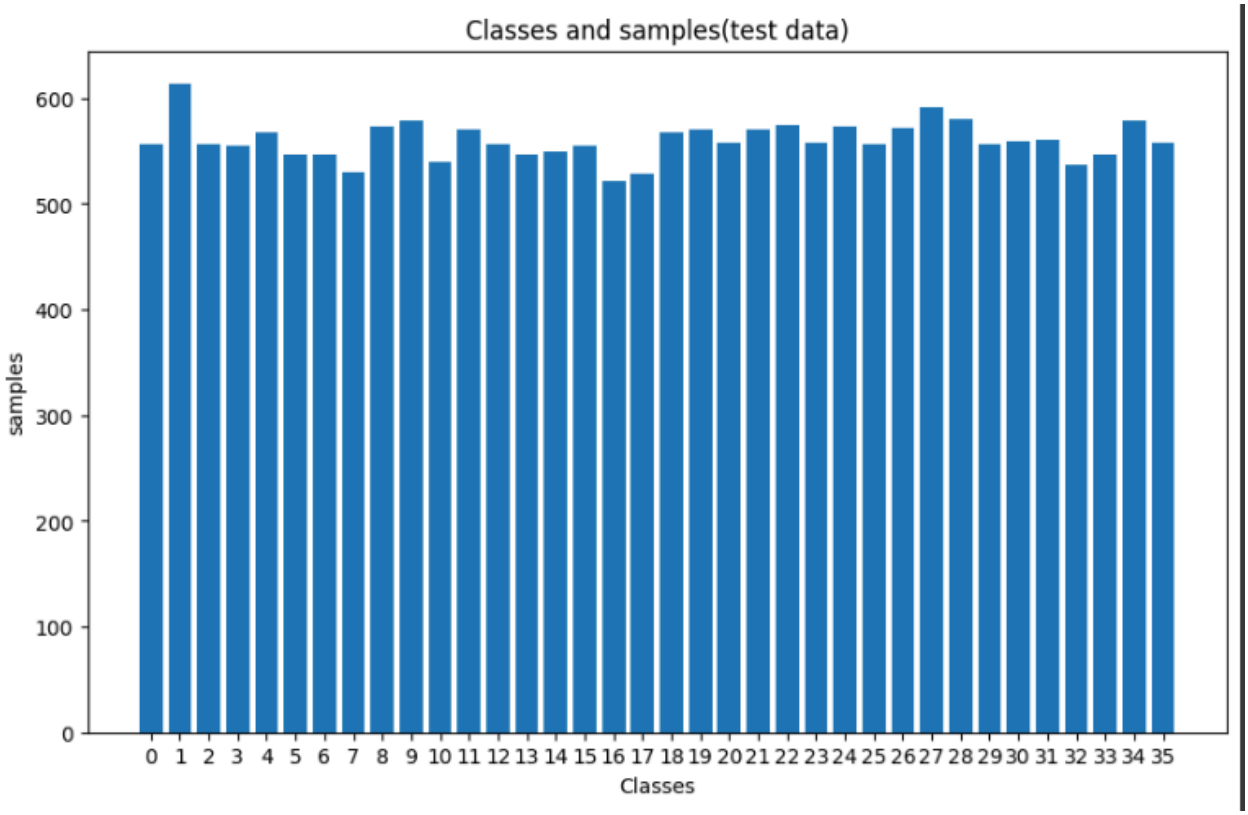
- The bar graph above shows the information of classes and samples provided in the dataset.
- As mentioned above in the overview of the dataset we have 36 distinct classes which are 0-9 numbers having 10 classes and a-z alphabets having 26 classes, the above graph shows each class and exact number of sample images in it.
- Each of the 36 classes shows a sample size to be 2800.

B.



- The Graph above shows information of classes and samples in it , but this is on the training data which we split into 60% to training and remaining to test and validation data.
- So 60% of the total samples in each class comes close to 1680 , which is also reflected in our graph.

C.



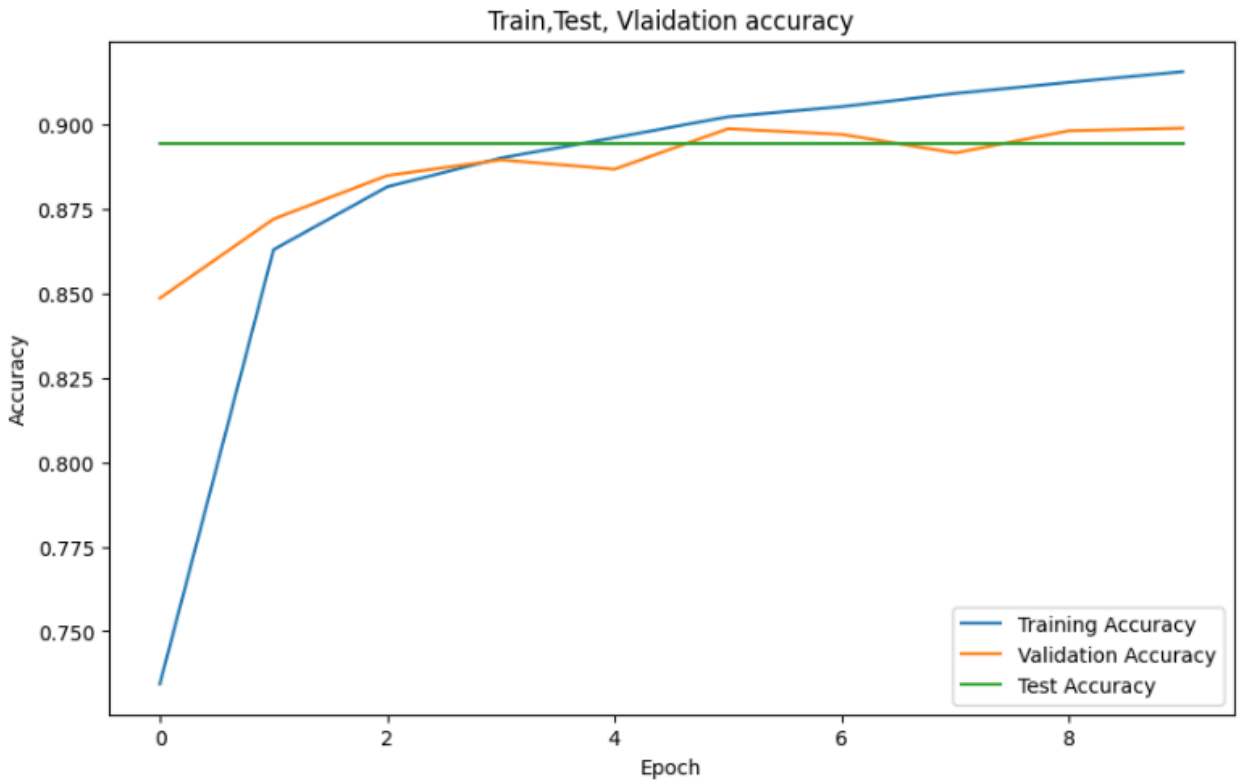
- The Graph above shows information of classes and samples in it , but this is on the test data which we split into 20% to testing and remaining to train and validation data.
- So 20% of the total samples in each class comes close to 560 , which is also reflected in our graph.

3. Provide a summary of your final CNN model that returns the best results:

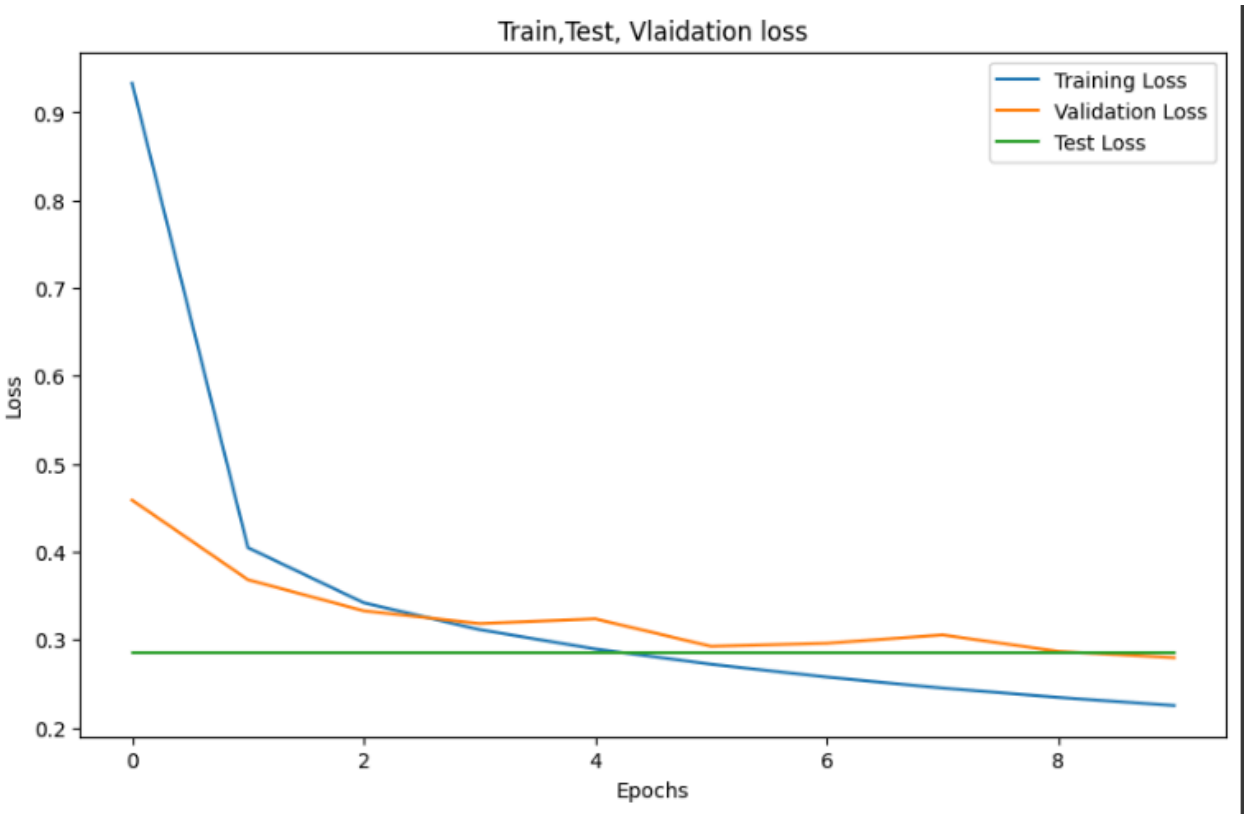
- We have transformed all images to grayscale that is single channel to make it less complex for training.
- We have used 2 convolution layers and an input layer passing images of dimension 28X28 and a single channel.
- We have received the best accuracy when we implemented batch normalization in our base model.
- We have used two 2D convolution layers as the kernel moves by 2 dimensions on the data, followed by a kernel of size 3 and padding =1.
- Followed by the first convolution layer we have added a batch normalization having 16 channels.

- To this we have used ReLU as an activation function.
- We have followed the same for the second convolution layer only difference is it has 32 filters and output dimension after this layer would be halved by that of output of first layer i.e. 14x14.
- The output dimension of the first layer is 28x28 which will be halved in the second layer as mentioned because of the maxpool layer added.
- The maxpool layer reduces the dimension by half of its original size.
- And a second pool layer would further reduce it to a 7X7 dimension.
- We then use view() to flatten the image to 1 dimension so it would be easy to train, so the dimension from the 2nd max pool layer would be flattened to 1D vector.
- Then 2 fully connected layers help to reduce the dimensionality of these feature maps to a lower dimension.

4. Provide performance metrics and graphs:

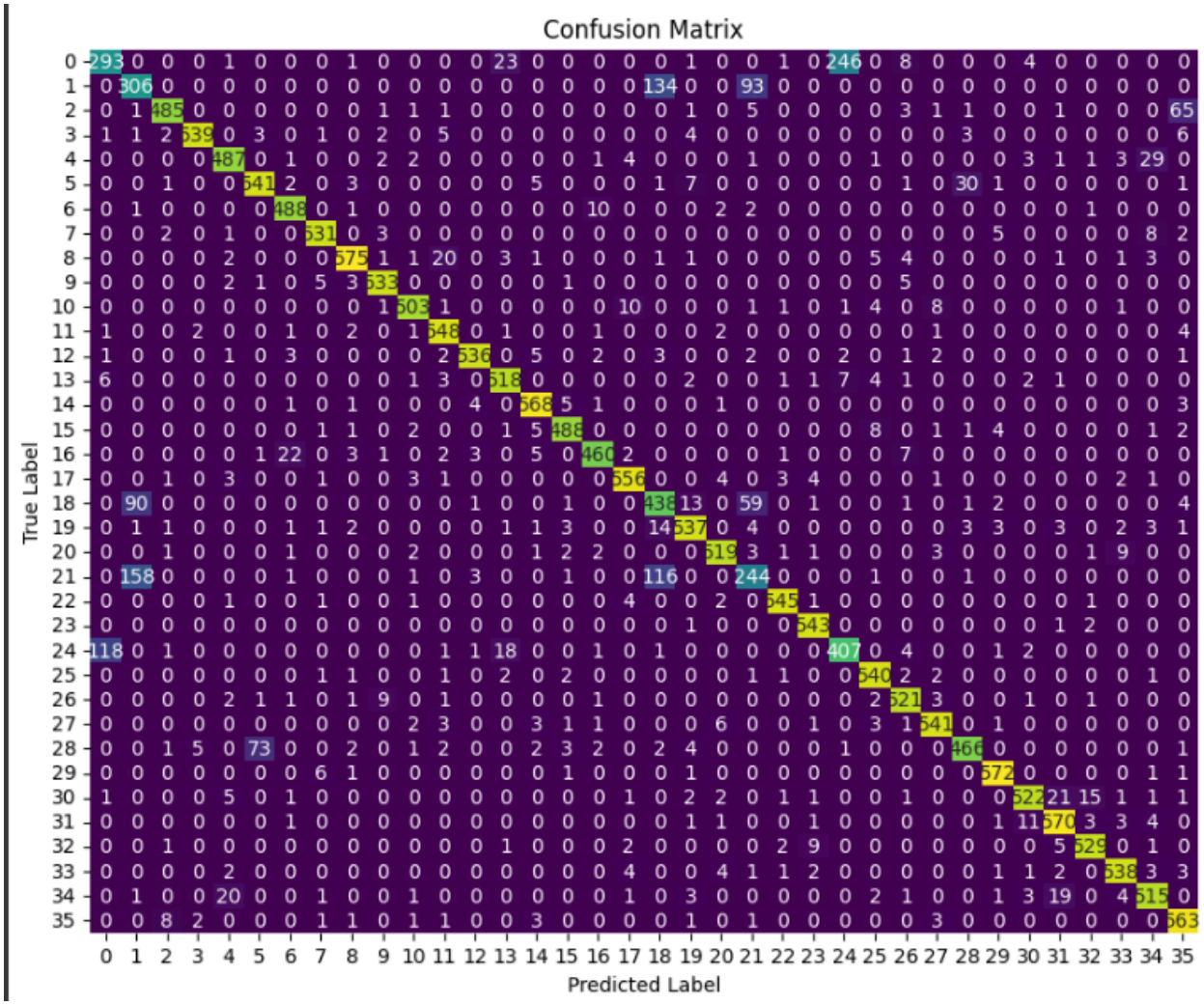


- The above graph compares the accuracies we have got while training a model on training , testing and validation data.
- We can observe in the above graph the test accuracy seems pretty constant at around 89% but train and test accuracies are variable until a certain point.
- The training accuracy has improved from almost 75% to above 90%, the maximum training accuracy we have received while training data in our best model i.e Batch normalization is 91.58%.
- The validation accuracy also seems to be improved from 85 % and maximum accuracy we received in the batch normalization model is 89.90%.

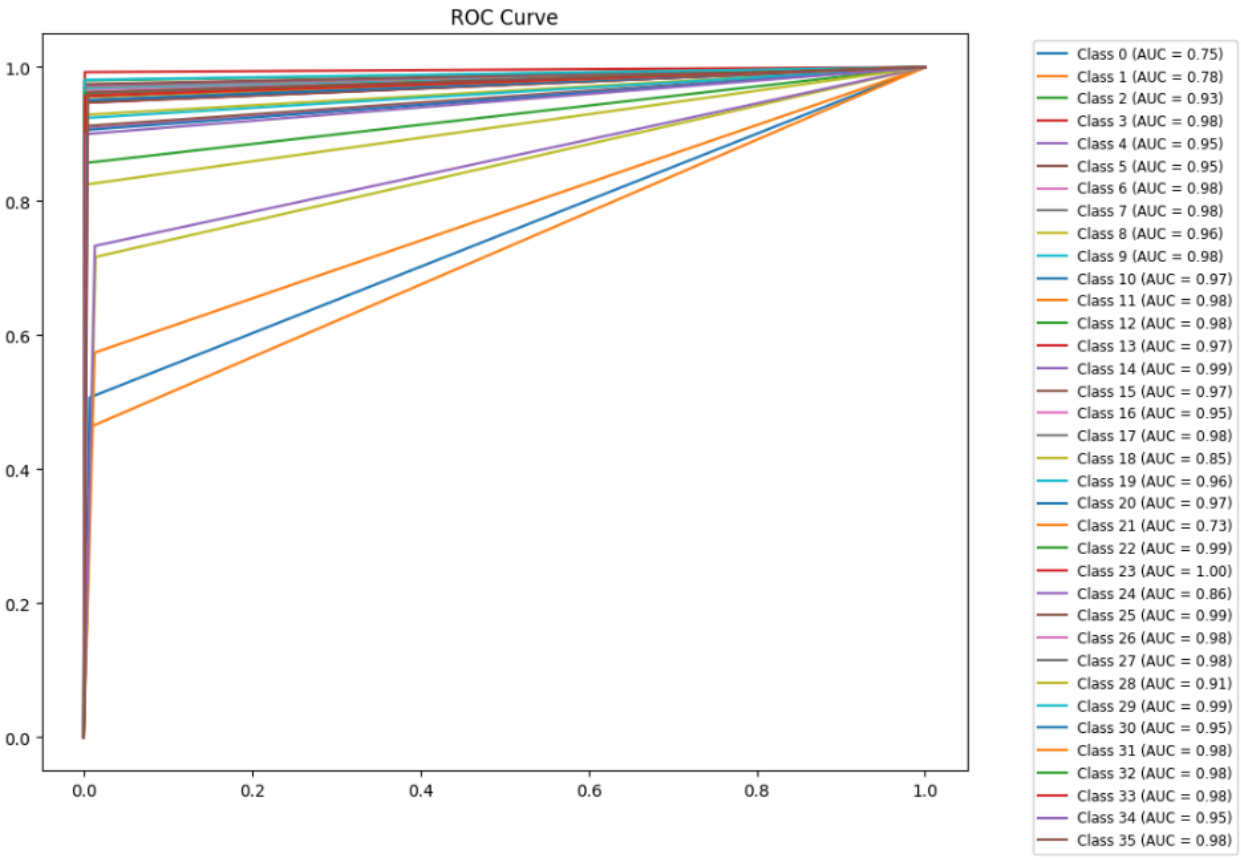


- The above graph compares the loss values we got during the training of the model on all train, test and validation set of data.
- The training loss seems to be higher in the beginning, almost above 1% gradually decreasing and getting lowest value around 0.2.
- The test loss seems to be constant between 0.2 to 0.3 as we are running a model on test data which is 20% of total data and it is unshuffled.

- Validation loss also lowered from near to 0.5 to almost lowest near to 0.3, we have the same amount of data as test data for validation also which is 20% of whole data.



- We can observe above confusion matrix as we have 36 classes we can see its predicted values and true values.
- If we see class zero it has 293 predictions correct class 35 has 563 correct predictions.
- Diagonally each class has a pretty good number of predictions 293 to 563.



- The ROC curve above shows the True positive values have more values as it is mostly curved or cornered to an end.
- All the classes compared differentiated with true positive or false positive values model given when ran on a set of data.

PART 4

VGG-13 Model

Since the images used for training on this model are 28X28 whereas the VGG-13 model requires an input of size 224X224, we have initially transformed and resized the images to fit the model input requirement. We have again transformed all the images to grayscale to make them less complex. This architecture allows us to easily train the provided EMINST dataset by resizing the input images to fit the model input requirement and utilizing all the 13 layers of the model.

In our model we have utilized 13 layers to process images which have been resized to 244X244 along with a single channel.

We have utilized 5 convolution layer blocks each containing two convolution layers with varying filter size, kernel size of 3 and padding set to 1. Between each convolution layer the RELU activation function is used.

In convolution block 1, we have layers with 64 filters and a maxpool layer which reduces the size of the image in half i.e. to 112X112. For convolution block 2, we have followed the same structure, only doubling the filters to 128 and reducing the maxpool layer by half to 56X56. For each consecutive convolution block we have increased the filters by doubling each time and reducing the maxpool layer by half until our final pool layer dimension of 7X7.

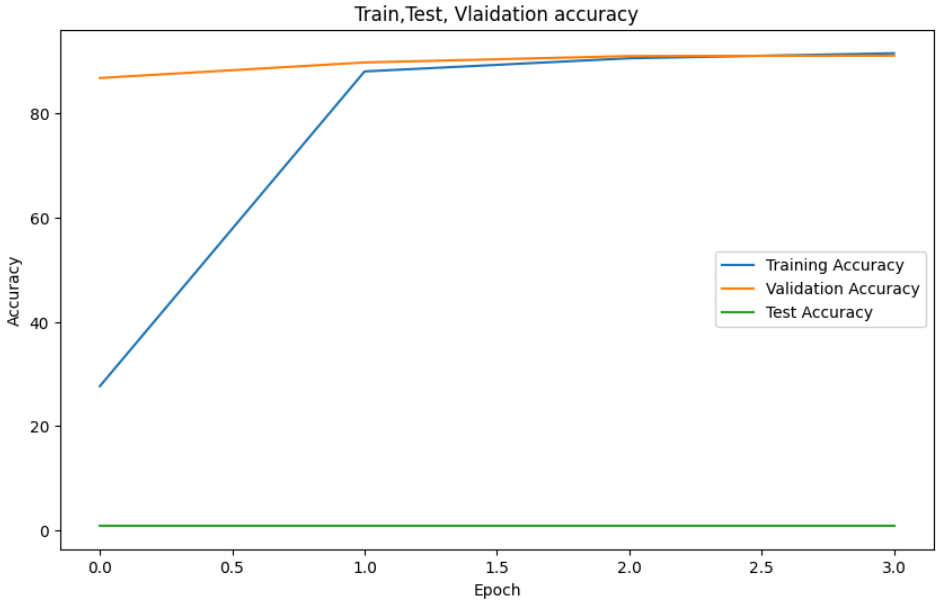
After our convolution layers we have our output received from the final maxpool layer flattened from 512 X 7 X 7 feature matrix to a one dimensional vector. After which we have our first fully connected layer which has 4096 neurons. After this we have our RELU activation function and dropout function. The second fully connected layer is also of 4096 neurons which is then again followed by RELU activation function and dropout layer.

Architecture:

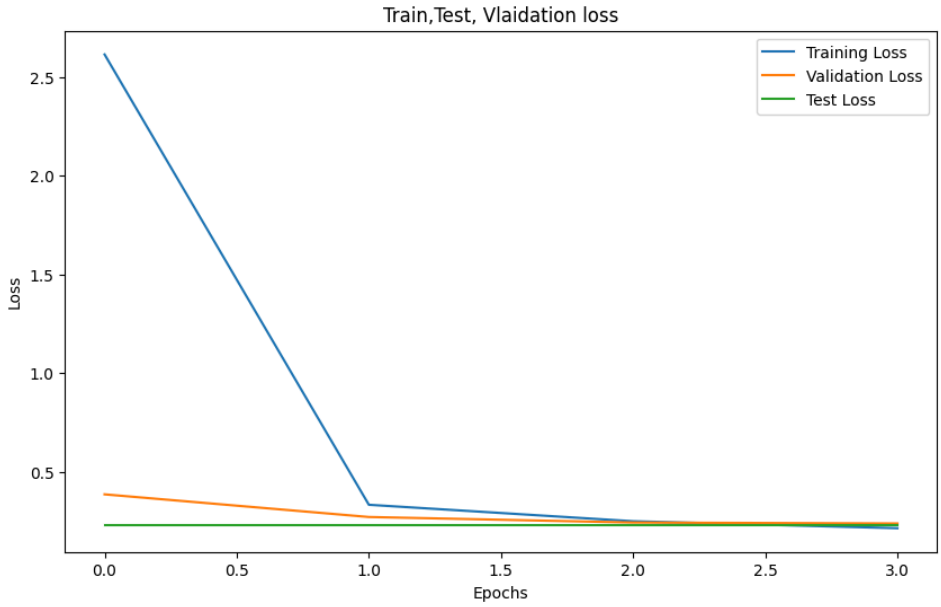
In part III we used 2 convolution layers and also the most optimized model with

maximum accuracy received is when we applied batch normalization on it. It has two 2D convolution layers as the kernel moves by 2 dimensions on the data, followed with a kernel of size 3 and padding =1. Followed by batch normalization having 16 channels. Same we again implement for the second convolution layer followed by batch normalization. We have used ReLU as an activation function in it. Also we have also used 2 max pool layers which reduces the complexity of the model by reducing the dimension by half.

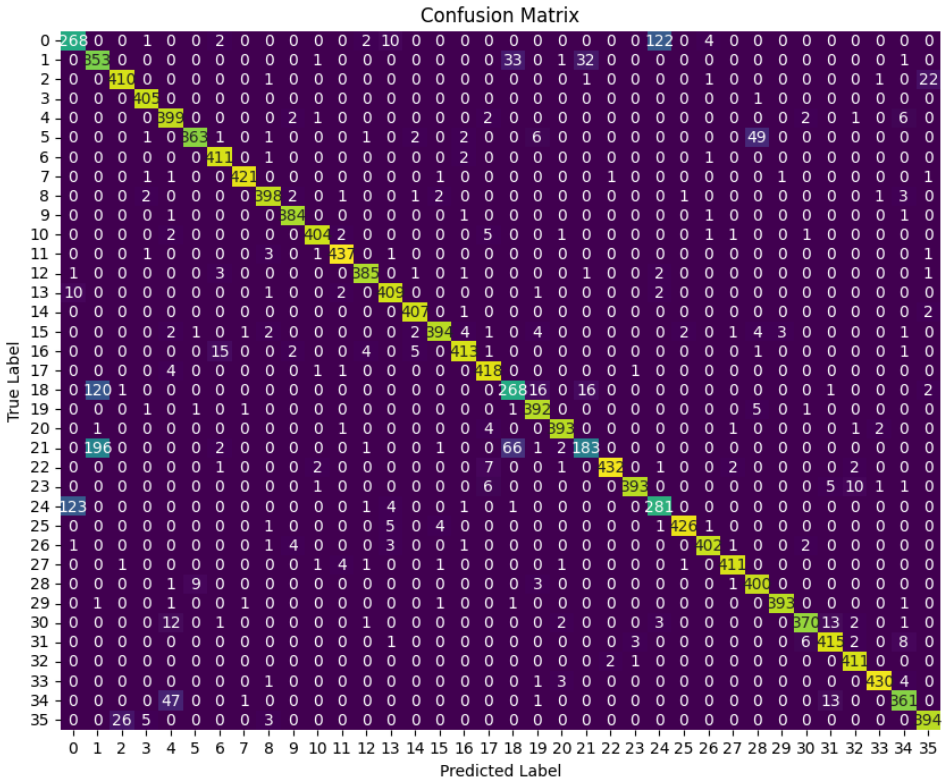
Result Analysis



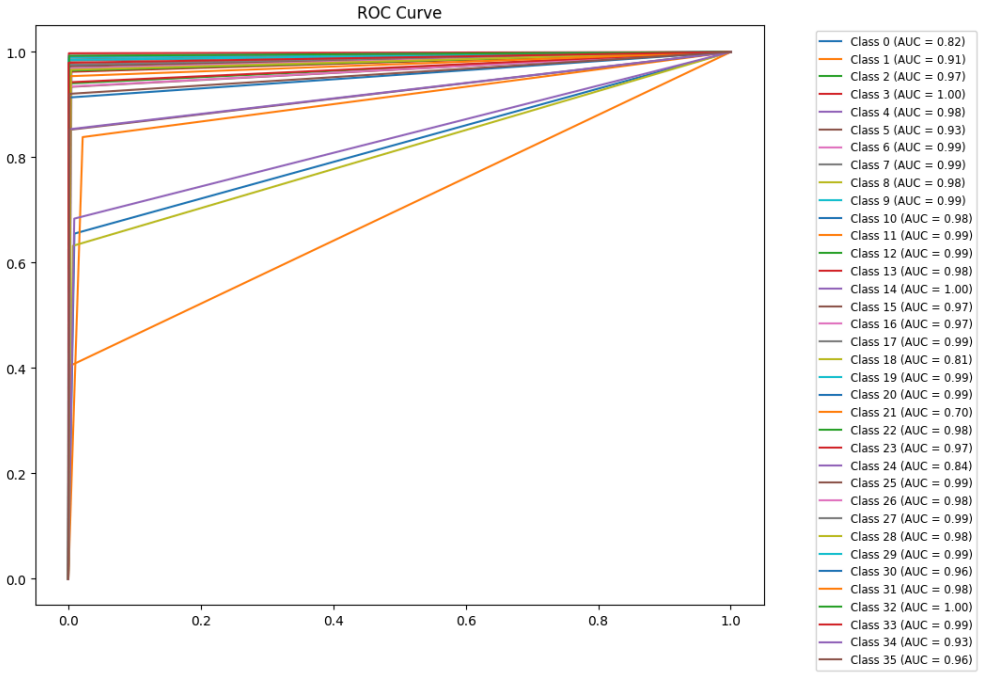
- The above graph compares the accuracies we have got while training a model on training , testing and validation data.
- We can observe in the above graph the train and test accuracies are variable until a certain point.
- The training accuracy has improved from almost 30% to above 90%, the maximum training accuracy we have received while training data in our best model which is 91.55%.
- The validation accuracy also seems to be improved from 85 % and maximum accuracy we received for the model is 91.09%.



- The above graph compares the loss values we got during the training of the model on all train, test and validation set of data.
- The training loss seems to be higher in the beginning, above 2% gradually decreasing and getting lowest value around 0.2.
- The test loss seems to be constant between 0.2 to 0.3 as we are running a model on test data which is 15% of total data and it is unshuffled.
- Validation loss also lowered from near to 0.5 to almost lowest near to 0.3, we have the same amount of data as test data for validation also which is 15% of whole data.



- We can observe the above confusion matrix as we have 36 classes and we can see its predicted values and true values.
- If we see class zero it has 268 predictions correct and class 35 has 394 correct predictions.
- Diagonally each class has a pretty good number of predictions 183 to 426.



- The ROC curve above shows the True positive values have more values as it is mostly curved or cornered to an end.
- All the classes were differentiated with true positive or false positive values model given when run on a set of data.

REFERENCES:

1. https://pytorch.org/docs/stable/generated/torch.optim.lr_scheduler.ReduceLROnPlateau.html

2. https://pytorch.org/docs/stable/tensors.html

3. https://pytorch.org/tutorials/beginner/basics/buildmodel_tutorial.html

4. https://pytorch.org/docs/stable/nn.html#loss-functions

5. https://pytorch.org/docs/stable/generated/torch.nn.BatchNorm2d.html#torch.nn.BatchNorm2d

6. https://medium.com/@vrunda.bhattbhatt/a-step-by-step-guide-to-early-stopping-in-tensorflow-and-pytorch-59c1e3d0e376

7. https://scikit-learn.org/dev/auto_examples/model_selection/plot_roc.html

Contribution

Team Member	Assignment Part	Contribution
Dhiraj Patil	Part I, Part II, Part III, Part IV	50%
Anirudh Mhaske	Part I, Part II, Part III, Part IV	50%